APPLICATION

FOR

UNITED STATES LETTERS PATENT

TITLE: DISTRIBUTED COMPONENT SYSTEM MANAGEMENT

APPLICANT: ANTHONY L. TOIVONEN

"EXPRESS MAIL" Mailing Label Number EL253774158US
Date of Deposit 10/29/99
I hereby certify under 37 CFR 1.10 that this correspondence is being deposited with the United States Postal Service as **"Express Mail Post Office To Addressee"** with sufficient postage on the date indicated above and is addressed ~~to the~~ Assistant Commissioner for Patents, Washington, D.C. 20231

Rich Donovan

PATENT
10559/043001/P7397

DISTRIBUTED COMPONENT SYSTEM MANAGEMENT

Background

The invention relates to distributed component system enhancement.

The explosive growth of the Internet, the increasing popularity of personal computers, and the advances in high-speed network access have brought distributed computing into the main stream. To simplify network programming and to realize component-based software architecture, a distributed computing system called a distributed component object model (DCOM) has been introduced. However, current remote activation mechanisms afforded by the DCOM are inadequate in servicing machine-independent requests. Although a mechanism is available within the confines of a single machine to request an identity independent implementation of some interface, such as in component categories, no integral facility exists for querying the network as a whole for any available host capable of servicing a request independent of host identity.

Sub A1

Since DCOM activation requests depend on the client being aware of the name and/or IP address of a specific server, the client often has either to monitor the detailed state of the network at the time of the request or to assume the server is available and configured properly to service the request. Server

failures become evident to the client only after the activation request has been committed **to the server by the client [CORRECT?]**, at which time it may be too late for the client to mitigate the problem. There is often no mechanism available for the client to dynamically attempt connections with other anonymous and viable nodes in response to a failure of the currently used server because of the static nature of a DCOM based distributed system. At best, a response to the server failure often requires informing application users based on the network configuration, and at worst, it may require a complete recompilation of source code.

end A'

Summary

A distributed component system in which component object model (COM) clients can create components at run-time without knowledge of the specific names or capabilities of the network nodes servicing that creation is disclosed.

The components in the system are distributed among a client node and its server nodes. The system includes a client augmentation module to intercept and process client activation requests, and server augmentation modules monitoring activation requests from the client node. The client augmentation module processes the activation requests by multicasting the specifics of the request to the network. The server augmentation modules

allow the client node to create remote components on the available server nodes without monitoring a detailed state of the network.

Other features and advantages will become apparent from the following description and drawings, and from the claims.

Brief Description of the Drawings

FIG. 1 is a block diagram of a distributed component object model (DCOM) architecture;

FIG. 2 shows an enhanced DCOM framework;

FIG. 3 is a block diagram of an IP augmentation module for the client node;

FIG. 4 is a block diagram of an IP augmentation module for the server node;

FIG. 5 is a flow diagram of the IP augmentation module for the client node; and

FIG. 6 is a flow diagram of the IP augmentation module for the server node.

Detailed Description

A basic requirement of a distributed system is an ability to create components. A block diagram of the distributed component object model (DCOM) architecture, shown in FIG. 1, defines how components 102 and their clients 100 interact over a network.

The network can be a local-area network (LAN), a wide-area network (WAN), or the Internet.

In object-oriented programming and distributed object technology, a component 102 is a reusable program building block that can be combined with other components in the same or other computers in a distributed network to form an application. Components can be deployed on different servers in a network and communicate with each other for needed services. Examples of a component include an interface to a database manager and a single button in a graphical user interface (GUI).

Component object model (COM) run-time libraries 104 provide object-oriented services to clients 100 and components 102. The libraries 104 use Distributed Computing Environment (DCE) Remote Procedure Call (RPC) 106 and security provider 108 to generate standard network packets that conform to the DCOM network-protocol 112 standard. The packets are placed on a protocol stack 114 before being shipped to the network.

In the COM environment, object classes are named with globally unique identifiers (GUIDs). When GUIDs are used to refer to particular classes of objects, they are called Class Identifiers (CLSIDs). The CLSIDs are 128-bit integers, used in RPC systems like DCOM, which provide a decentralized namespace for object classes. For a DCOM environment, the object creation mechanism in the COM libraries 104 is enhanced to allow object

creation on other machines. In order to be able to create a remote object, the COM libraries 104 need to know the network name of the server. Once the server name and the CLSID are known, a portion of the COM libraries called the Service Control Manager (SCM) 110 on the client machine connects to the SCM on the server machine and requests creation of the object.

When indicating a remote server name at the time of an object creation, the DCOM allows clients 100 to maintain location transparency. That is, clients 100 need not know whether the component 102 is running locally or remotely. Therefore, when the remote server name is made part of the server component's configuration information on the client machine, clients 100 do not have to maintain or obtain the server location. All a client needs to know is the server name and the CLSID of the component. It simply calls 'CoCreateInstance,' and the COM libraries transparently create the correct component on the server machine.

An enhanced distributed component object model (DCOM) framework 200 that augments the existing activation capabilities of the DCOM service is illustrated in FIG. 2. The enhanced DCOM framework 200 takes the location transparency of the standard DCOM one step further by allowing the clients to maintain transparency in the name or capability of the network node servicing the request. Thus, a client in the enhanced DCOM framework 200 does not need to know even the server name of the

component and the client can process activation requests in a machine-independent manner.

An augmentation/enhancement to the standard DCOM creation mechanisms 210 comprises two parts: a first part 202, called an Internet Protocol (IP) augmentation for the client node, intercepts and processes client activation requests; and a second part 204, called an IP augmentation for the server node, monitors requests on the server machines 208.

The IP augmentation for the client node 202 intercepts the standard client activation request and broadcasts the specifics of the request to the network 212. The request is for a list of server IP addresses or universal naming convention (UNC) names of servers that have the ability to service a request for a specific CLSID or for an interface via a CLSID directly. The IP augmentation for the server node 204 monitors a specific port that is tied to a multicast IP address. Depending on the mode of operation, the IP augmentation 204 can simply return the server IP address or use the standard DCOM creation mechanism to create, package, and return an interface pointer in a location transparent form.

A block diagram of the IP augmentation module for the client node 202 is shown in FIG. 3. The augmented DCOM creation mechanism 202 includes multicast enhancements that intercept and process client activation requests from the standard DCOM

creation mechanism 210.

A block diagram of the IP augmentation module for the server node 204 is shown in FIG. 4. The augmented DCOM creation mechanism 204 includes multicast enhancements that monitor requests on the server machines.

FIG. 5 shows a flow diagram of the IP augmentation module for the client system 202. The diagram illustrates a process by which a COM client 206 triggers the creation of components at run-time without knowledge of the specific name or capabilities of the network node servicing that creation request.

The process begins when the augmentation module 202 intercepts standard client activation request, at step 500. The augmentation module 202 then broadcasts the specifics of the request to the network, at step 502. This broadcast is sent to an IP multicast address (step 504) and a specific port that is configurable at a system maintenance level (step 506). The mechanism by which this broadcast takes effect is entirely implementation dependent and has no direct impact on the system at a higher level.

The client 206 may request component activation using two different modes of operation, at step 508. In a Server Name Request (SNR) mode (step 510), the client makes a request for a list of server IP addresses or UNC names of servers that have the ability to service a request for a specific CLSID. In a Multi-

CoCreateInstance (Multi-CI) mode (step 512), the client bypasses server naming and requests an interface via a CLSID directly.

In the SNR mode, the client 206 provides a CLSID, an Interface Identifier (IID), a maximum and minimum response wait time, a maximum and minimum response count, and whether server names or IP addresses should be returned. This type of request will result in the return of one to many server names or IP addresses capable of servicing a DCOM activation request for the particular CLSID and IID requested. Once the client has these server names, it can then proceed as normal using the standard DCOM mechanism for remote activation.

The primary advantage of the SNR mode to the client is the flexibility it affords. A client can make intelligent decisions prior to a DCOM activation request based on what it now knows, or can discern, from the list of available servers it received. For example, a known unreliable or failure prone server could be dropped in favor of a different server from the list.

In the Multi-CI mode, the parameters for the client request include a maximum response wait time as well as maximum and minimum response count just as with the SNR mode, but the returned values will instead be the interface pointers requested. The IP augmentation module for the client node 202 creates location independent references to objects on the network by using an existing DCOM protocol known as an Object RPC (ORPC)

[<-- WE SHOULD DESCRIBE THIS PROCESS IN DETAIL. I TRIED TO WRITE OUT THE PROCESS (TWO PARAGRAPHS BELOW; IT MAY NOT MAKE SENSE). PLEASE ADD, CORRECT OR DELETE AS YOU SEE FIT]. The ORPC is a set of definition that extends the standard DCE RPC protocol. It specifies how calls are made across the network and how references to objects are represented and maintained.

ORPC uses standard RPC packets, with additional DCOM specific information, in the form of an interface pointer identifier, conveyed as additional parameters on calls and replies. The interface pointer identifier is used to identify a specific interface on a specific object on a server machine where the call will be processed.

One of the parameters of an activation response packet is the marshaled interface pointer which is represented in an object reference (OBJREF) structure. The OBJREF structure is a data type that represents a reference to an object and contains a signature field of hex value 0x5747454D. This sequence which reads 'MEOW' in ASCII is useful when scanning for the object reference packet.

A flow diagram of the IP augmentation module for the server system 202 is shown in FIG. 6. The system 202 monitors and listens on a specific port that is tied to the multicast IP address, at step 600. Again, the server may service the

component activation request by the client using two different modes of operation, at step 602.

In the SNR mode, the server machine 208 simply returns the server's own machine name or IP address to the client (step 604). In the Multi-CI mode, when the broadcast activation request arrives, the IP augmentation module for the server machine 202 exercises the normal DCOM creation mechanisms to attempt to service the request (step 606). If this response creation is successful (step 608), then the resulting interface pointer will be packaged into a location transparent form and sent to the client for use (step 610). Specifically, a DCOM remote OBJREF in the form of a MEOW packet, described above, can be used to send the requested information to the client. Finally, the augmentation module for the server machine 202 retrieves the client address from the IP packet information (step 612) and sends the requested information to the client (step 614).

The above augmentations or enhancements to the existing functionality afford the server machine 208 the ability to implement intelligence regarding the requests it will respond to and those it will ignore. By responding only if the response creation is successful, the process prevents network reconfiguration or recompilation of source code. Further, since the reception of broadcasted activation requests is tied to a specific port, the configuration of the available network servers

can be modified to result in different replies for clients that initiate a broadcast request.

Some of the advantages of the enhanced DCOM framework 200 which offers a generalized activation scheme include easy implementation of virtual DCOM servers and flexible organization and distribution of the server components that can be dynamically altered without the client systems being aware of the changes. The virtual DCOM servers are comprised of several physical server nodes on the network.

Other embodiments are within the scope of the following claims. For example, a client could broadcast a request to the network for a specific CLSID which would result in the eventual return of 1 to N marshaled COM class factory interface pointers. Also, a client could specify a specific CLSID/IID pair and receive some number of marshaled interface pointers that correspond to the unique IID requested. Furthermore, the client can broadcast a desired COM category identifier used for generalized creation, and subsequently receive some number of interface pointers without regard to any particular underlying implementation or identity.